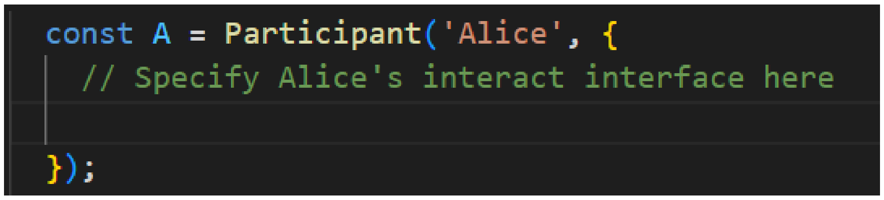
API Participants Challenge

API Participants are a special type of Participant in Reach. They allow us to repeat users with the same functionality. So far, you may have seen Alice defined as a Participant like this:

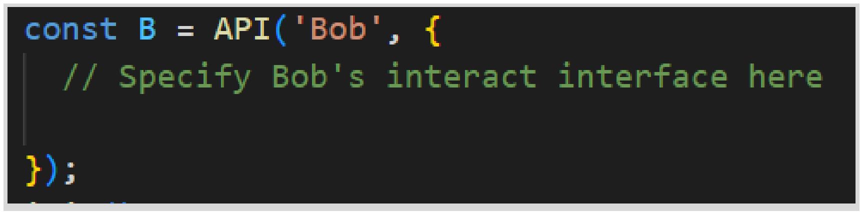


```
const A = Participant('Alice', {
  // Specify Alice's interact interface here

});
```

This allows one address for Alice to have the defined custom functionality specified by the interact interface.

If we want to allow repeat users for Bob, instead of defining the user as a Participant, we would define them as an API Participant. We define API Participants like this:



```
const B = API('Bob', {
  // Specify Bob's interact interface here

});
```

These are essentially a user type that shares the same functionality. Instead of defining a Participant for each different user, Reach allows us to repeat an API Participant when a new user attaches to the contract.

Imagine an application where users vote for their favorite color. There can be unlimited voters who participate in the act of voting. Each voter is an independent user, but they all only need to vote. In this example, all voters are members of the voter API. In this program, I’ll refer to these API Participants as Bobs.

General Challenge Description:

There are 3 levels to this program. **You can only submit for one level**. Payouts are denoted for each level – you will only receive payout for the level of program you are submitting. (i.e. submit level 2 get $40 USDC)

We recommend building level 1 first and then moving on to level 2. If you are having trouble with level 2 you can fall back on your level 1 solution for submission.

Building a level 3 solution for each of the Hack Challenges is a great way to grow your Reach project portfolio.

Plagiarism Warning

Plagiarism weakens the community and does nothing to improve your web3 development skills. Submissions that are plagiarized will not be accepted.

These programs attempt to get you thinking about building code solutions from word problems. These should be your own work written from scratch.

If you are stuck, lost, or confused, reach out to us on Discord for assistance.

Program Requirements (Choose one level):

- **Level 1** (**$20**)
  - Write a program that allows multiple users to attach to the same contract.
    - Define one (1) Participant to deploy the contract
    - Define one (1) API participant
      - This allows multiple users to attach as Bob (Bobs)
    - Create an array for the new users
    - Create a function to reuse when creating new Bob users (Bobs).
      - This should include newTestAccount
      - This should include attaching the account to the backend
        - Be sure to attach to Alice's contract information
      - Add only the address of each account to your users array
    - This should be done in 2 files
      - index.rsh and index.mjs
    - Console messages
      - Include a message when Alice is ready to accept Attachers
        - Alice should interact in an only block with this function
      - Include a message when you are creating new Bob users
      - Display your users array

- **Level 2** (**$40**)
  - Use a parallelReduce
    - parallelReduce is a key abstraction in Reach. It is essentially an abbreviation of a while loop combined with a fork statement that you write yourself.
  - When a new API user attaches, increment a counter of attachers
  - Allow participants until there are 5 in your users array
  - Console messages
    - Display a message when Alice is ready to accept Attachers

        - This should be called from the backend
        - Include a message when you are creating new Bob users
        - Display a message each time you increment the counter
        - Display a message that the user tried to attach but the max of 5 attachers has been reached

- ➔ **Level 3** (**$60**)
    - ◆ Build a frontend GUI
    - ◆ Use the framework of your choice
        - It will contain as many files as you need to complete the task
    - ◆ Use a View
    - ◆ Be creative on a theme for your DApp!

- ☐ Checklist
    - ☐ Does your program use an API Participant?
    - ☐ Did you provide solutions for all of the problems on your program level?
    - ☐ Are you submitting for just one level?
    - ☐ My submission is my original work